BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

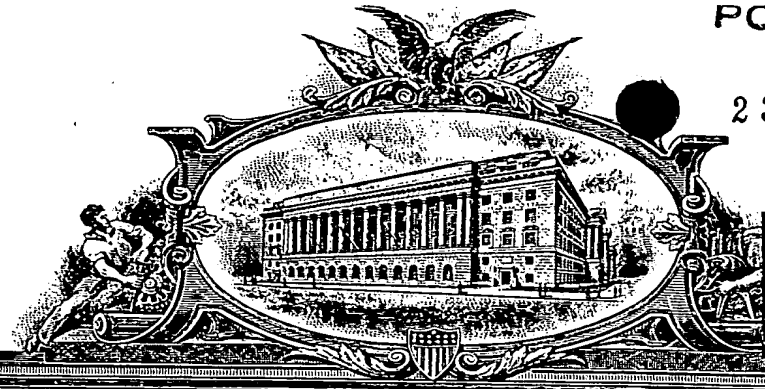
As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

M.H

PCT / CA 99/00873

23 DEC 1999 (23.12.99)

4



PA 185284

REC'D 06 JAN 2000

WIPO PCT

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE

United States Patent and Trademark Office

December 15, 1999

THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY FROM THE RECORDS OF THE UNITED STATES PATENT AND TRADEMARK OFFICE OF THOSE PAPERS OF THE BELOW IDENTIFIED PATENT APPLICATION THAT MET THE REQUIREMENTS TO BE GRANTED A FILING DATE UNDER 35 USC 111.

APPLICATION NUMBER: ***60/101,857***
FILING DATE: ***September 25, 1998***

PRIORITY DOCUMENT

SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)

By Authority of the
COMMISSIONER OF PATENTS AND TRADEMARKS

N. Woodson

N. WOODSON
Certifying Officer

Approv

VISIONAL APPLICATION COVER SHEET

This is a request for filing a PROVISIONAL APPLICATION under 37 CFR 1.53(c).

09/25/98 JC614 U.S. PTO

Docket No. 02310.6000	Type a plus sign(+) inside this box ->	+

INVENTOR(s)/APPLICANT(s)

LAST NAME	FIRST NAME	MIDDLE INITIAL	RESIDENCE (CITY AND EITHER STATE OR FOREIGN COUNTRY
SNELGROVE	Martin		
STUMM	Michael		
DESIMONE	Mauricio		

09/25/98 JC614 U.S. PTO 60/101857

TITLE OF THE INVENTION

NETWORK OPERATING SYSTEM FOR TELEPHONY

CORRESPONDENCE ADDRESS

FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, L.L.P.
1300 I Street, N.W.
Washington, D.C. 20005-3315
Telephone No. (202) 408-4000

ENCLOSED APPLICATION PARTS (check all that apply)

[xx]	Specification	Number of Pages [45]	[ ]	Small Entity Statement	
[ ]	Drawing(s)	Number of Sheets [00]	[ ]	Other (specify)	_____

METHOD OF PAYMENT (check one)

		PROVISIONAL FILING FEE
[XX]	A check or money order is enclosed to cover the Provisional filing fees.	_X_ $150.00
[ ]	The Commissioner is hereby authorized to charge filing fees and credit Deposit Account Number [_____]	___ $ 75.00 (small entity)

The invention was made by an agency of the United States Government or under contract with an agency of the United States Government.

[XX] No.

[ ] Yes, the name of the U.S. Government agency and the Government contract number are:

_____

Respectfully submitted,

SIGNATURE _____ Date: September 25, 1998

TYPED OR PRINTED NAME Ernest F. Chapman Registration No. 25,961

[ ] Additional inventors are being named on separately numbered sheets attached hereto.

PROVISIONAL APPLICATION FILING ONLY

WST's Network OS for Telecom

Network Operating System for Telecom

Martin Snelgrove, Michael Stumm, Mauricio De Simone

A draft white paper overview of WST's proposed NOS, to be used for internal project definition and as an initial draft for patent applications.

WST is developing a telecommunications system with two key but distinct components: wireless connectivity and an open software architecture. This document focuses on the software, but includes brief descriptions of relevant hardware issues.

The two components combine to make a system that gives service providers access to end users (the wireless link) and a platform (the open OS) for developing a wide range of specialized voice and data services. Physically, the system consists of a collection of "patchpoints" in the end-user's homes or businesses, connected to a network of basestations through a third-generation (data-capable) cell-phone link. The patchpoints support voice (through RJ-11 jacks) and data (through an Ethernet connection).

The system is data-centric in that it uses a very general model of telecommunications, with voice services as a particular case, but voice is a key "use case" for which good performance is a must.

We are defining an applications programming interface (API) which will permit third-party developers to develop new call-processing and filtering software. The business advantages are that we punch above our weight with the help of third parties and that they can adapt to market niches that we wouldn't see. The key technical requirements are that the API be very secure and very clean, and these are advantages for our own developers too.

The network operating system is distributed over the entire collection of patchpoints and basestations. Key requirements in designing it are scalability, performance and reliability, all obtained with modern distributed-OS software techniques such as the use of narrowly-defined servers, efficient message-passing, and process migration.

In contrast to conventional telephony systems, which involve interconnecting several types of specialized computers with predefined functions, we have undifferentiated hardware resources that are assigned tasks dynamically. This reduces design costs for us and inventory and engineering cost for service providers.

We define a "middleware" layer that abstracts the key elements of communications (connectivity and quality of service) from the detailed implementations in IP[1], ATM, frame relay or circuit-switched networks. This allows our

developers to make applications that are portable and future-proofed. We connect to the PSTN and other networks at gateways.

Our hardware is based where possible on PC hardware, to get high-volume consumer costs. Network interfaces (e.g. T1) are available off-the-shelf, but in order to get ahead of the pack on the critical wireless link (we're using a standard due to be finalized in the year 2000 and not expected to ship in volume for cellular telephony until 2003) we're developing a custom modem board for it. This board will be designed as bare-bones but flexible hardware ("RISC philosophy") and will implement only the physical layer, with some flexible hardware support for the link layer.

The wireless link standard we are working with is the FDD version of the ARIB wideband CDMA proposal. A key problem in CDMA is that Qualcomm charge high licence fees, and there is a business risk that the final standard relies on their technology. In this case we will support both proprietary and standard versions, with the proprietary versions winning on cost and the standard versions on compatibility. Our modem hardware will be flexible enough to work either way.

1.0 The Physical Network

Figure 1 describes our network from a physical point of view, with basestations connected to a network "cloud", illustrating several types of connectivity that we should be able to handle. A simpler picture would just connect all basestations together through IP, but that would cost us the ability to exploit the strengths of ATM and of point-to-point or LAN connections between our basestations. That could limit our ability to deliver high quality of service.

Maintaining the freedom to use different types of network, and the ability to track the changes in IP, will involve our defining a "middleware" layer that expresses networking at a more abstract level than allowed by ATM or IP. In particular we will have to do a careful job of defining the trade-offs between quality of service and price. These connectivity and quality negotiating definitions will be key parts of our API.

The basestations are rugged commercial PCs with specialized cards, such WST radio cards, where necessary. The standard platform gives us the economies of scale of the PC industry. The patchpoint will use PC-market chips, but will have to fit in a small footprint and will have to have a very low unit cost, so it will be a (simple) custom design. An option we should consider is making a PC-card (perhaps PCI) version of a patchpoint.

The wireless link will be based on the emerging "3G" (third generation cellular radio) standard, which is specified to be able to provide data rates up to 2Mb/s in fixed applications such as ours, though that rate may hog an unacceptable amount of bandwidth in a typical system. Section 4.0 outlines important properties of the wireless link.

"Power users", who want more bandwidth than we can economically provide, may be wired to our system and still be able to use our software: the wireless and open-system aspects of WST's product are distinct.

1.1 DSP Hardware

We need to support signal processing at MHz rates in the radio modem and at audio rates in telephony coders and fax modems. A traditional architecture would have the MHz processing done in an ASIC, the audio-rate work done in a specialized DSP, and the control in a microprocessor.

1. We'll use "IP" for Internet Protocol and "IPR" for Intellectual Property (Rights), to avoid ambiguity. The main form of IPR we're focussing on at the moment is the patent.

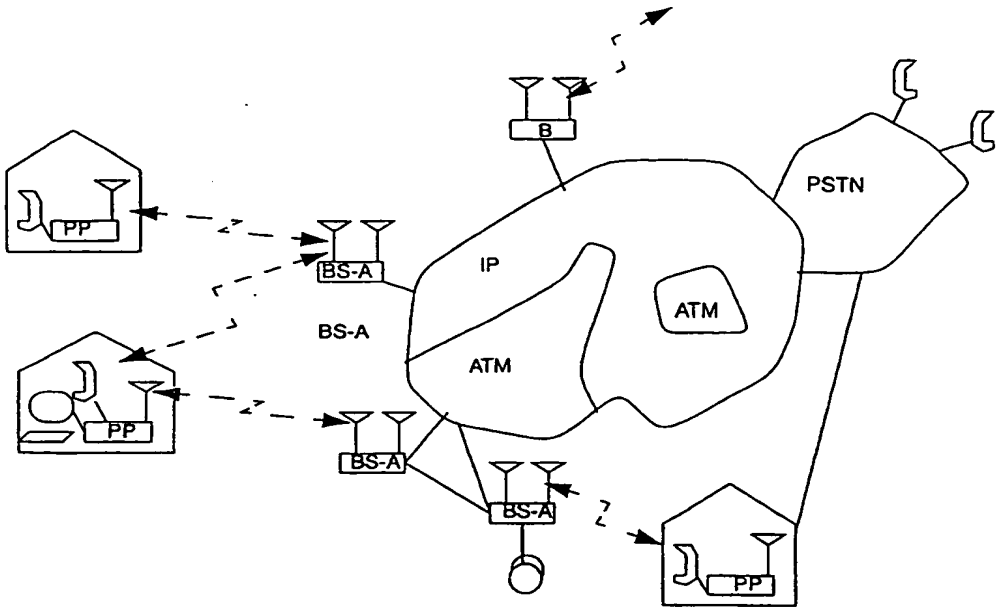


FIGURE 1. Service provider "A" has a network of WST basestations ("BS-A") that relies on IP and ATM routers to provide a backbone; users have patchpoints ("PP") connected to basestations with redundant illumination where possible. WST or other gateways connect to the public switched telephone network and other proprietary networks; users store voice-mail on rented disks; and the network software collaborates where possible with WST equipment owned by other providers such as "B".

We need quite a flexible radio, because 3G standards are evolving and handle multiple rates, so some programmability is needed in the modem. It will initially be implemented on an FPGA, which is a conventional first step in ASIC development, but the final design will look like an FPGA specialized to signal processing. This may be capable of handling some of the audio signal processing as well as radio-modem functions, and will be programmed in VHDL (probably with some constraints to make synthesis easier). There may be value in partnering with Morphics on this, if they have anything real to contribute, and we should have someone FPGA-savvy on the radio team. Making VHDL code something that can safely be loaded from a library and attached to a user's process is desirable, since it maintains the full signal-processing flexibility that we are aiming at.

The microprocessor will be a high-powered consumer part, and these have special support for computer graphics (e.g. the MMX instruction set) that we may be able to retarget for audio DSP. A DSP specialist will be needed to evaluate the capabilities of these machines for filtering and modem functions. A partnership with Gao Research may help, since they are implementing standard modems in Pentium software.

It is not clear that we will need a conventional DSP, and avoiding using one will reduce the complexity of our software maintenance effort, particularly as time passes and we end up with several generations of hardware. We should include one only after it is proven that we can't use the FPGA or CPU to do a vital job.

2.0 Three Telecom Issues

Broadly, telecommunications involves switching, billing and provisioning. Every packet through the network involves all three issues, and making a good system will require us to deliver on all three. Third parties will write call processing software that configures the system for their data, and that must always set up the system so that all three are handled consistently. In an open network it is not necessarily in the best interests of a software developer that billing be handled properly, and we should not expect the developer to understand provisioning issues, so maintaining consistency at the same time as offering flexibility is a key requirement on our middleware.

Switching involves moving data around the network and processing it as necessary on the way. A natural view of this is that call processing software defines a graph whose nodes process data and whose edges process it, with subscribers as particularly interesting nodes. We will often leave a lot of the hard work to the backbone, which would typically contain Internet or ATM routers. We will need to be able to describe a wide range of performance parameters of processing nodes and of links so that calls can be set up properly. This will be the topic of Section 5.0.

Billing is traditionally thought of as a matter of filing durations and destinations of calls for postprocessing, but we need to think this through at a deeper level so as to be able to extend readily to new payment technologies (such as e-cash) and to allow for third-party software to take over functions like detailed billing and exclusion of expensive calls. We also need a consistent philosophy for billing packet data and voice, because the difference is only really defined at the user level in our open system. We also have to make pricing and billing information available to the user's call processing software, and should institute sophisticated price negotiation mechanisms so that system resources are used efficiently even with third parties setting up calls.

"Provisioning", or more generally Operations, Administration, Maintenance and Provisioning (OAM&P) code gives the network operator the information needed to operate. We need to report failures and load statistics to an operator's terminal (probably a website, in practice) so that repairs can be scheduled and decisions on new equipment purchases can be made on sound technical and business bases.

3.0 Quality of Service

We are mixing voice and data, both on the wireless link and the service provider's backbone, and carrying several types of data at that. Requirements are different for different applications, and unless we want to be stuck in the low-quality market we'll focus our system software on respecting those needs. At present voice-over-IP is low quality, with long and variable packet delays; the traditional IP method to solve that is to offer excess network capacity, but that is expensive and anyway not compatible with a system in which traffic is carried over third-party networks. We should not rely on the IP infrastructure to solve our QoS problem.

We will be implementing a system in which contention for resources is handled by user processes bidding for them, whereas IP networks have traditionally implemented centralized policies aimed at ensuring some type of fairness and have guessed at priorities based on such things as source and destination addresses and type of traffic.

Given finite capacity on communication links in a network, there are two ways to give some packets priority: by preferential treatment in the queues to get on links (multiple queues or different likelihood of packets being dropped) or by choosing routes (where redundant ones exist) so that high-priority packets travel over lightly loaded links. ATM and advanced IP systems use these strategies internally, and we will need to control them.

Data transmission, such as for file transfer and Web browsing, tends to have high requirements on bandwidth and accuracy, while for voice calls low latency is a much more important concern — it is famously difficult to hold a

conversation over a channel with a 100ms delay. Statistical properties of voice and data are also quite different, with data transmission being burstier and likely to use TCP. (which penalizes the network for dropping packets under load by retransmitting them, which can cause a type of instability) Bandwidth is expensive because a given network only has so much of it; low latency is expensive because it restricts the use of queues in the transport network, which are needed to smooth out statistical variations in loads.

As new types of service emerge, the balance of requirements will be different. For example, video-conferencing fundamentally has higher data rates than voice but the same latency requirements. At the moment it is very poorly handled: typical video-conferencing equipment uses heavy compression to get the bandwidth down to what can be handled with an ISDN line, and thereby adds so much delay that conversation is difficult — and the picture is lousy to boot. If we offer the performance and price required to make high-quality videoconferencing economic, the volume of this type of traffic may build up quickly — as fax did. There may also be a hidden market for high-quality audio in telephony, either in specialized areas (like radio broadcast) or as a matter of price for the general public, and this would also tighten specifications. These new markets could offer our customers a real edge, so we have to be able to offer them easily. We can't design the system so that it only works well for the current service mix.

The telcos have revenue models based on making money from voice, and the rapidly decreasing cost of bandwidth puts them in a quandary: they can't afford to drop prices near costs, and until they do the new services that might make money won't emerge. Half of "voice calls" are now actually faxes, which also wastes resources because the application doesn't fundamentally require (expensive) low latency, and so businesses are emerging which bypass the telco for fax. The standard interface provided by an RJ-11 jack and a line-card is limiting their ability to engineer their system.

Subjective quality is the real measure of interest in a telephone system, and is usually quantified with an "MOS" (mean opinion score) by having a collection of users rate various connections on a scale of 1-4. The service provider is faced with costs that are functions of error rates, bandwidths and latencies and an ill-defined MOS value model.

Telephony has traditionally controlled load by call admission — refusing business that exceeds the volume the system can handle (although the effect of refusing to carry a call is that the caller dials again, so that at peak times the "call attempt" load goes up). The Internet Protocol model is "best effort", so that when load increases speed suffers, (for everyone, at the moment) rather than using call admission. In IP, you don't set up a call, you just send the packet. Internet audio is moving to a model in which the coder changes as a function of network load, using a bandwidth-efficient coder when loads are high and a better-sounding one when the capacity is there. This is radically different from the call-admission model, and probably much more acceptable to the end-user — a mechanical-sounding call on Christmas day is better than spending an hour hitting "redial". This is an area in which our customers can beat the telcos.

IP has been extended to handle voice latency requirements with "type of service" bits which can be used to assign priorities in switching, "tags" which can force precomputed routes rather than risking delays for routing computations, and a reservation protocol (RSVP). These features are not supported by most of the current infrastructure, and even with them a fair amount of network engineering is required to provide a given latency. In the Internet world, there is a trade-off between network engineering and raw bandwidth, in that a network with enough excess capacity will be fast. Microsoft is apparently pushing RSVP hard, because Windows 98 uses it, so we can expect a lot of RSVP traffic from the Ethernet port. Unless they add billing support, though, their demands for priority won't be respected by carriers. We can add the missing piece.

ATM (Asynchronous Transfer Mode) networks are an attempted compromise between the needs for data and voice. They use a call admission model, and you have to set up a route before sending data, but they can handle bursty

traffic. They specify a complicated collection of traffic models, so applications can in principle define their needs and the network can in principle make guarantees, but these features are seeing limited use.

Our system has to support IP and voice applications, and on IP, ATM, voice and other networks. If we have to model ourselves on just one protocol, it should probably be an up-to-date IP. We are probably better off, though, to interpose a layer of middleware that allows us to define a model that we feel addresses the real problems but that maps easily onto the existing traffic and networks; this will allow us to adapt to innovations in IP and new applications, and to make billing and operations respond to the market's real needs.

Leaky buckets are used both in ATM and RSVP to specify average bandwidth. Traffic is modelled in terms of the average output rate and the size of the input buffer needed to smooth bursts out to that rate. A long burst will overflow the bucket, and packets that overflow the bucket are typically marked as candidates for deletion if the network overloads. For the radio link we might interpret these parameters literally, allocating enough radio slots/channels to handle the rate, and putting a buffer at the sending side; for an optical link it may be interpreted only as a specification that defines which packets may be marked for sacrifice. A variant (inverted?) mechanism is a "token bucket" that allows bursts at full speed until the flow has used up a bucket full of tokens, then restricts flow rate to the required average as tokens dribble in. These mechanisms directly express queueing behaviour, which is fundamental to networking, so they seem to be the right ones to use.

The choice of what to do with overflow packets is typically fixed in present systems, with packets marked as candidates for deletion, but our system should be flexible enough to allow us to define a wide variety of policies—such as backpressure mechanisms.

For coded voice, the average rate is about 50% (the voice activity factor) of the peak, but users would want to allocate enough bandwidth for the peak so that monologues don't get delayed in a buffer. The radio system still benefits from low voice activity, though, because interference is reduced. (Section 4.0 below) A model for 8kb/s coded voice might be a token bucket (don't delay data) with an input rate of 8kb/s, refilled with tokens at 5kb/s (a little margin over 50% utilization) and tens of seconds deep (so that it doesn't empty for 99% of speech bursts). The right thing to do on overrun will vary according to desired voice quality and whether there is competing traffic: for example the price could go up, a lower-rate coder could be substituted, or a greater FER accepted.

The leaky bucket model doesn't necessarily tell us everything that we need to know in setting up a path: in a packet-switching system we will generally have internal queues whose length is a function of aggregate traffic, and we will want to know how the sources interact. We may need to develop a more informative model, but it will have to degenerate easily to the leaky bucket, because that is what both ATM and RSVP use. One example would be to use a collection of buckets to describe average rates when measured at a variety of queue sizes; a generalization would be some mathematical function describing the relation between queue length and expected rate; and yet more general would be a set of functions relating queue length to a collection of rate statistics (mean and variance, or a collection of percentiles). We shouldn't expect typical developers to be able to figure these things out, so we will probably want to develop profiling tools that can do the work for them.

IPR later?

3.1 QoS in ATM

Asynchronous Transfer Mode sends 48-byte payload packets ("cells") over prearranged routes ("circuits"). The standards include sophisticated mechanisms for defining and guaranteeing quality of service at the time of setting up a circuit, but they are little used because of their complexity and because of jurisdictional problems (applications have no reason to be moderate in their demands). Our challenge is to use pricing to control greed, and to make an API that gives the QoS control promised by ATM in a comprehensible way.

ATM defines "service classes" A through D and corresponding "QoS classes" 1 through 4. This is a fairly coarse set of categories. "Service categories" are also defined, which are parametrized so as to allow expression of a wide range of needs and service guarantees. These are what we're most interested in: the user specifies the statistical profile of the traffic and in return the network guarantees a certain performance level. There are needs to police traffic (to check that it fits the promised profile) and shape it (to make sure that it does).

Constant Bit-Rate (CBR) is straightforward in definition, setting bandwidth with a given peak cell rate (PCR). The user also defines the tolerable cell delay variation (CVDT), which he/she expects to smooth out with buffering at the destination. Cell transfer delay (CDT), cell loss ratio (CLR) and peak-to-peak cell delay variation (CDV) are specified by the network as its QoS guarantee. Traditional toll-quality telephony needs this service category, but it's wasteful. Users would expect to pay by the minute.

Real-time Variable Bit Rate (rt-VBR) allows bursts up to a peak cell rate and maximum burst size (MBS) and guarantees cell transfer delay (CTD) and its tolerable variation (CDVT) at a specified sustainable cell rate (SCR). A leaky bucket can be used to police or shape traffic to this profile. This is more like what we need for modern telephony. Users would probably still expect to pay by the minute.

Non-real-time VBR doesn't guarantee CTD. It's probably more like what a Web browser would want. Users would likely expect to pay by the minute, but would probably want a discount for slow packets.

Unspecified Bit Rate (UBR) actually does specify peak cell rate, but not a sustainable cell rate. It's basically best-effort. Users might expect to pay by the megabyte.

Available Bit Rate specifies a minimum cell rate (MCR) as well as the peak, and the network uses back-pressure to control the flow: the network sends "Resource Management" cells to the source to allow it to adapt to the capacity available. Users might expect to pay by the minute for the minimum cell-rate and to pay a slight premium when rates are high, or (equivalently from a mathematical point of view, but not psychologically) to pay for premium service but get a discount when forced back down to the minimum rate. This mechanism should be able to get the best utilization out of the network when users have sophisticated rate-adaptive coders, and might be a winner for videophone.

3.2 QoS in IPv6 and up

- IP often uses ATM internally
- priority and class
- tag switching
- Integrated Services Architecture

RSVP (the resource reservation protocol) is a good start: a packet goes from sender to receiver setting up a path, and another comes back reserving resources (which has to be repeated every 30 seconds). The path is supposed to be set up with enough priority to get the effect of a lightly loaded network. Traffic statistics are specified by the parameters of a leaky bucket, and packets beyond the stated rate are marked as eligible for deletion. The mechanism gives us only two levels of service (high and ordinary priority).

IP priorities are implemented differently in different subnetworks, and often not implemented at all.

3.3 Cost and price of QoS

- "fixed cost" at timescale of a call, hence congestion pricing

- provisioning "unfixes" the cost, mechanism for overall system to respond to loads.
- provisioning motivation for provider, vs. letting congestion drive up price: competition puts ceiling on price; increased volume tracks lower prices at higher capacity.
- provider needs to set policies about customer satisfaction/ % potential demand satisfied, then we help implement them.

3.4 Warranty mechanism

Between failures and statistical variability, QoS cannot be absolutely guaranteed; the standard commercial response to that problem is to offer warranties that specify how failures to meet requirements will be handled — MOS 3.5 or your money back for the last 5 minutes, for example. This is a particularly important mechanism for defining pricing for events that are inherently statistical, because it offers a natural way to define whether or not performance conformed to specifications. For example, if the network offers 1000 packets free of charge for every one dropped, it had better keep its packet loss rate below 10^{-3}

IPR?

Traffic and network performance are in fact only described statistically, and this is at the heart of the difficulty in providing QoS: statistics of real traffic are subtle and not easily expressed in a few parameters — how long is a typical voice burst? How long when the caller is a poet? The science of statistics was developed in order to model betting, and a bet is the natural abstraction of a complicated set of statistical measures. Money-back mechanisms are essentially bets between the network and the subscribers.

4.0 About the Wireless Link

The open system should be useful with or without wireless links, but for business reasons we must support them early. Bandwidth on these links is expensive to the service provider, and must be used efficiently. This section is a quick overview of aspects of the wireless link that are critical to the software design.

Radio channels are not like "pipes", because their signals spread over a wide area rather than being carried point-to-point and they thus interfere with each other. This means that there are system-level interactions between users and that radio resource allocation is a subtle art. For example, link error rates can be reduced by increasing transmitter power, but this increases interference levels for other users and increases their error rates; the overall system can easily become unstable. WST software has to give us the ability to use advanced techniques in this area and to track innovation.

CDMA (code division multiple access) is a technique for allowing multiple users to share a piece of radio spectrum. We're going with it on a business guess. There are two key variants, one out of Qualcomm and one that attempts to break Qualcomm's stranglehold on relevant IPR. We'll go for the second, but with the flexibility to change quickly if needed.

CDMA channels are defined by a sequence of 64, usually, "chips" (fractions of data bits) with values of ± 1 ;[1] each channel is orthogonal to each other, and bits are transmitted by sending the given sequence multiplied by ± 1 .[2] Channels

1. Actually, complex values are used, and ARIB transmits control on the "imaginary axis" and data on the "real". We may want to change that decision to get more usable bandwidth.
2. Another area where we may want to innovate.

have equal maximum bandwidth, namely the chip rate (4.096M/s, for ARIB CDMA in 5MHz) divided by the code length (64, in this example, for a data rate of 64kb/s per channel)[1].

Channels may have different *average* bandwidth, though, in that system interference is set by the number of channels actually in use, rather than the number allocated. Thus a source like voice that only transmits about 50% of the time uses less system capacity than a file transfer that transmits continuously. In IS-95, a system might assign only 30 of its 64 channels, and rely on voice statistics to mean that on average only 15 of them are in use, and rarely 30. A data user doing file transfer would use almost twice as much of the system capacity even though assigned a single channel.

high FER CDMA wireless links are typically operated with frame error rates that are high (10^-3) relative to those over optical and wired links (<10^-9). Since this means that most errors will occur in these links, they are typically protected with local error correction that can solve the problem immediately, rather than leaving it to an end-to-end mechanism like TCP. For voice, forward error correction and power control are used to maintain an acceptable error rate, rather than an ARQ mechanism that would add latency. Generally some bits need to be protected more carefully than others, so we would like users to be able to manipulate error control in some detail.

cell packing and SIR vs. local BW In principle it is possible to increase transmitter power and thereby force error rates arbitrarily low, but then the transmitter interferes with other users nearby; thus best cell packing density is obtained by deliberately operating at a marginal signal to interference ratio (SIR). CDMA standards generally have quite sophisticated power control to optimize overall network capacity.

Unreliable links can be dealt with by forward error correction (error-correcting codes) or by automatic repeat request. ARQ wins if there is no latency constraint, as in file transfer, but for voice and video it is better to get fresh data than to waste time on that lost and hence moderate FEC wins.

power control is a technical issue that is particularly important in CDMA systems. Multipath propagation of radio signals can cause interference between channels in CDMA, where it cannot in FDMA[2], and so CDMA systems are limited in capacity by interference. The two ways to mitigate this are through careful power control, so that no-one transmits any more power than they have to, and advanced digital signal processing techniques that cancel some of the interference.

Power control is done with a combination of open-loop (if the signal you receive is strong, then the channel is good and you can transmit at low power) and closed-loop (the other party periodically asks you to turn your power up or down) techniques. The need for closed-loop control means that the radio modem itself is involved in sending and receiving very simple packets on a millisecond timescale (0.625ms for ARIB) as part of every communication.

use of random-access, paging and allocated channels. Radio channels may be used in an Aloha style (like Ethernet, with packets all sharing a channel and occasionally colliding and needing to be resent) or may be allocated to particular users. Both styles are valuable: the Aloha (or "random access") style for small volumes of traffic (small relative to a channel) and allocated channels for larger volumes (because Ethernet only gets a utilization of 1/e in heavy load). In the 3G standard, as in cellular systems generally, requests for service (dialling) are carried on random access channels, while voice traffic is carried on dedicated data channels.

We can (and should, according to 3G) transport IP either way according to volume. Strategies for choosing between them can be based on traffic statistics (allocate a channel when the collision backoff gets excessive, say, or when a

1. For further complexity, ARIB divides each 10ms frame into 16 timeslots, so a data channel is really 1/16 of this
2. In FDMA systems, multipath causes fading instead.

queue length exceeds a threshold) but may sometimes be easier to specify at the application level — bandwidth demands of voice and video are relatively well understood. Only our users know whether a given data stream represents voice or data, so they will have to (directly, or indirectly through their requests for QoS guarantees) specify the choice.

Radio channels may also be multicast. In a mobile network, "paging" is used to alert a subscriber to an incoming call, and pages are sent out by multiple basestations because the subscriber's location in uncertain. This is not multicast at the logical level, because only one receiver needs to hear the page, but all of them monitor the channel. A clearer use of multicast is that the basestation broadcasts the interference levels that it is suffering ("you'll all have to speak up a little, it's noisy up here"); another is that "soft handoff" (see below) involves having a patchpoint in redundant communication with two basestations at once, so that either can be dropped without loss of the call.

Some carriers use a type of multicast (for dispatch and similar push-to-talk applications) at the service level (e.g. Nextel and Clearnet's Mike service), but this doesn't necessarily get implemented by sharing radio channels — it just overlays a virtual private network on the mobile network.

Multicast in the CDMA environment makes the problem of closed-loop power control tricky, because enough power has to be used to communicate with the most distant receiver — which would have to be identified as the proper source of feedback. Paging and other broadcast (in the radio sense) messages are therefore transmitted at full power. The need to do this reduces the bandwidth saving available from multicast. A similar problem arises if an ARQ technique (e.g. TCP) is used.

IPR?

For WLL, we may be able to identify the "most distant receiver" reasonably consistently and hence allow multicast without overkill on power. If there is enough traffic (preferably of a type that doesn't need ARQ) to make the resulting bandwidth saving significant, WST should innovate in the use of multicast radio channels.

short codes vs. multiple codes: BW quantization and multicast (?) vs. hardware complexity at the modem

Telephony systems were designed for a constant-bandwidth service, but data services need bandwidth on demand. A single CDMA channel, with rate 1/2 coding for a 10^{-3} FER, has a data capacity of 32kb/s[1] and can be divided into 16 time slots that can be used efficiently with 50% voice activity; so data rates from 32kb/s down to about 1kb/s can be accommodated in slots of a channel. At very low bandwidths the Aloha-style channel is best, and at higher rates multiple CDMA channels have to be used.

There are two strategies for transmitting on multiple channels, and both are legal in the ARIB proposal: the use of multiple codes and the use of shorter codes; WST will have to make a decision, and it's a hardware/software/capacity trade-off.

The use of multiple codes is logically straightforward: for 384kb/s, say, 12 modulators with different codes could be used and their results summed. One problem is that this puts 12 times the hardware into the patchpoint modem.[2] The second problem is that the radio signal has more dynamic range and a better power amplifier will be needed.

The use of shortened codes is more subtle: the 64×64 Hadamard matrix of codes has a binary tree structure that means that there are shorter codes of length 2, 4, 8, ..., 32 that are orthogonal to the rest of the codes: a code of length 16 allows us to transmit 4 bits of data during the 64-chip interval of a single code. The receiver is actually

1. less overhead, and also note that the "downlink" gets twice the bitrate of the uplink, but spends half of it on control.
2. The basestation has to handle this anyway, since it may be dealing with 40 or so CDMA channels per antenna.

simpler for the shorter code, and while the transmitter must operate at a higher power (proportional to bitrate) it doesn't have an increased dynamic range (peak/average) requirement over the single channel. The problem is that bandwidth can only be allocated in power-of-two multiples of the basic rate, and a 384kb/s connection (for example) would take up 16 slots. Another problem is fragmentation: as codes are allocated and given back, the available codes will be scattered at random, rather than bunched together (all of the leaves below a particular node of the binary tree) where they can be replaced by a short code.

WST is probably better off with the shortened codes: data services are so bursty that fine control of their bandwidths is likely meaningless, and unused capacity is not entirely wasted (as exploitation of the 50% activity factor for voice demonstrates) if the transmitter is shut off when not needed because system interference is reduced. The main penalty will be that we have to move channels frequently between codes to reduce fragmentation.

double illumination and handoff: relation to mobility, reliability and load-sharing

A patchpoint may often be "illuminated" by signals from two or more basestations, and this fact can be exploited in several ways. In mobile systems it has traditionally (IS-95) been used for "soft handoff", so that a mobile that is moving between two coverage areas can temporarily send its data to, and receive it from, both basestations. Then when it moves entirely out of range of one, the call can continue. Since the two paths can use different codes, this roughly doubles the workload of the modem.

During the period of double illumination, redundant packets are discarded, and when one is received in a corrupted state, the other is available. This obviously makes for a reliability enhancement, though at some cost in radio resource capacity, and WST may wish to allow fixed users to run in this mode when they are willing to pay for the quality.

Double illumination can also be exploited to allow load sharing between base stations (and is, in IS-95); when one is heavily loaded, traffic can smoothly move over to its neighbours. This economy of scope improves capacity.

licensed and unlicensed spectrum Our system can be operated in a variety of bands with fairly minor changes to the radio. The choice won't make any obvious difference to the software, except for interoperability (see below), but might affect performance statistics.

The two key bands for us at the moment are the 1.9GHz band, which is licensed, and the 2.45GHz band, which is unlicensed. The licensed band is expensive (the service provider may have to pay $1M and up for the license) but guaranteed free of outsiders adding interference.

At 2.45GHz there are spreading rules, (10:1 spreading, which would limit any end-user to an acceptable chiprate/10) and power limits (100mW, but a function of jurisdiction) that would restrict cell size or maximum bitrate (a few hundred metres at 32kb/s?). The 2.45GHz band also has a variety of interferers (principally microwave ovens and some wireless LANs, at the moment) and little control on who will enter in future, so a provider may run into trouble. This band is probably most interesting for campus-scale networks — perhaps a wireless PBX business.

opportunities for IPR vs. regulatory and interoperability, both at the physical and logical layers (ARIB)

Doing our own modem has two advantages: it guarantees that we'll have one when we need it, at a price we can predict, and it gives us a platform on which we can in principle innovate by modifying the ARIB standard.

IPR?

One example of an innovation is to generalize the QPSK/BPSK modulation that the standard applies to the spreading code (QPSK at 2b/symbol on the downlink, BPSK at 1b/symbol on the uplink) to QAM. This is generally not

desirable in mobile systems, because QAM signals get high bit-rate (6b/symbol for 64QAM, for example) at the cost of higher transmitter power, which in turn increases the radius over which other users are interfered with. For users close to the basestation, though, this is not an issue, and in a WLL system basestations can be placed close to power users, and in a system with flexible billing the additional capacity can be rationed by price in a manner sensitive to these radio resource issues.

At the next level of abstraction, the ARIB standard allocates logical channels to radio channels in particular ways, and in particular dedicates half the downlink capacity to control — which seems high and which throws away an opportunity to take some advantage of the asymmetrical up/down loads typical of Web browsing. We could also add more use of multicast and the option of allocating more Aloha-type capacity.

To innovate on the radio, we need to know how tightly regulation will constrain us and decide how much performance we're willing to sacrifice for interoperability with other people's equipment. The only interoperability we care about is at with other terminal equipment, e.g. mobiles, since we don't plan to hook our basestations up to other people's mobile switching centres. At the network end, we deal with other standards through gateways.

interoperability with IS-95

IS-95, or "narrowband CDMA" is a standard used for PCS cell-'phones, and has the same general mathematical structure as the wideband CDMA we are working with. It is possible to handle both systems at once, which is straightforward in different frequency slots and can even be done (with heavier signal processing and probably some reduced system capacity) in the same slot. Supporting PCS would be a selling point.

Qualcomm is pushing a W-CDMA standard that is closer to IS-95, which is theirs, in a few key ways. In particular, chip rates in IS-95 are multiples of 9600b/s, while in ARIB CDMA the base is 1kb/s. Qualcomm has a slightly lower chip rate overall, at 3(1.2288Mchip/s) = 3.6864Mchip/s as against 4.096Mchip/s for ARIB, but claim that their system will be cheaper to implement without spilling over into adjacent frequency bands. The Qualcomm basestations also rely on timing derived from GPS satellites, which Europe and Japan don't want to rely on and which doesn't work in buildings.

need for encryption

Analog cellular 'phones were embarrassingly easy to listen in on, and part of the advantage of the digital standards is that they take wiretapping out of the Radio Shack league. They are still probably not secure enough for business transactions (e.g. credit card transactions). What encryption there is in current commercial systems applies only to the radio link, with voice converted back to cleartext immediately in the network for compatibility with the PSTN.

We can allow users to write end-to-end encryption software, although there will be some interesting legal issues (with the FBI and NSA) if third-party vendors use keys beyond 40 bits. We also have to encrypt and authenticate our own control messages between basestations and patchpoints to avoid spoofing attacks.

4.1 FDD or TDD?

The 3G standards are supposed to support two modes for duplexing between up- and down-links: frequency-division duplexing, (FDD) which needs the paired frequency bands allocated for PCS, and time-division duplexing (TDD) which can work in a single band and for which new spectrum has been allocated.

There is a lot of commonality between the FDD and TDD versions, so that it will be possible to change course fairly easily if we start with the wrong one. Key points in favour of each are:

FDD
- is the technique used for PCS 'phones, and there's lots of underused licensed bandwidth out there
- PCS compatibility would in principle make it possible to turn on IS-95 compatibility in software
- peak radiated power levels are lower, (though W-CDMA has a TDM component that throws that advantage away) which will reduce power amplifier costs and may have EMC and even health advantages.
- doesn't have TDD's guard-time constraints, so synchronization may be simpler.

TDD
- doesn't need paired bands, so can use the unlicensed (ISM) bands at 915 and 2450MHz. The power limits in these bands would limit the use to microcells, and unlicensed bands are more vulnerable to interference.
- allows asymmetric allocation of bandwidth in the up- and down-links, which can be nearly 2x more efficient in spectrum utilization for applications like Web browsing.
- doesn't need a duplexer at the radio front end, for a savings of $2-$5.
- allows peer-to-peer networking, since there is no radio distinction between a basestation and a patchpoint. This in turn allows even more radical system innovation, and is a reason to go TDD in the long term.
- has up- and down-link channels that are perfectly reciprocal.(have the same transfer functions) This allows better power control (possibly reducing the penalty for avoiding Qualcomm IPR) and smart-antenna algorithms (which increase capacity)

5.0 Call Processing as Graph Design

Physically, the network can be seen as a graph with the edges being communications links and the nodes being the computers (patchpoints, basestations or compute servers); logically, it's natural enough to draw a similar network with the nodes performing computing functions and the edges still representing communication, but with computing functions possibly time-sharing computers and with the best implementation of the communications links being substantially different according to where computation happens. Call processing software "wires up" this logical graph, and maps it onto hardware.

Because the nodes in the logical graph can represent complex computing functions, such as voice coders, only certain interconnections are valid. These nodes have properties (such as latency and CPU load, for example) of interest when wiring them up. Because the edges carry different types of information, the nodes are best thought of as having typed "ports", such as (in the voice coder case) those for linearly digitized signals and for CELP-coded voice.

If edges on the logical graph represent physical communications links, they will also have properties (BW, latency, FER, ...). Edges and nodes are thus very similar objects, and so one approach would be to specialize them both from a superclass (GraphElement?); another approach would be to regard the physical link as a type of node with two ports (one for each end of the link), and so give it the same collection of properties as the computing node. In this case edges may have little meaning, and we basically connectPorts(coderA.out, FEC.in); we'll call this the Lego approach, where objects are connected directly at their ports with no explicit representation of the connecting edges. Choosing to maintain explicit edges at the logical level has the advantage that the logical graph may then be embedded directly in the physical one — which is an easy way of expressing the constraint that data has to get from one place to another. The "Lego" approach may be cleaner for pure computing applications.

Because our system is open, we have to distinguish between software that represents the interests of the end user, which is typically concerned primarily with the logical structure and only indirectly (through its effects on cost and performance) with the mapping onto hardware, and software that represents the interests of the service provider,

which is concerned with sharing hardware resources efficiently among a large number of users. This is a server/client relationship. We would prefer both types of software to be written in WST Java, but performance concerns might force us to compromise on the server side.

hierarchy or a grouping mechanism can be used to hide or show detail in the communication graph: it should be possible to connect(this, PSTN(555-1212)), for example, and implicitly create a useful connection with default coders and performance; it should also be possible to send an IP packet without needing to set up a connection. In the case of the 'phone call, it should be possible to descend into the hierarchy or group and see more detail — so that user code can add features, for example. Figure 2 shows some of the detail of signal flow in a typical digital wireless 'phone, with encryption inserted at a suitable spot; the kind of thing that we want third-party code to be able to specify. This code can then choose between end-to-end encryption and encryption only over the insecure wireless link, for example, enforce policies on what to do if the called party is on the PSTN or has forwarded the data to voice mail, and perhaps choose higher-performance error correction to compensate for the sensitivity of encrypted data to errors.

In the particular case of an IP packet, there may be no record of how the routing was implemented: so showing detail may not always be possible. It is not likely to be desirable to a user either, since the point of IP is to hide the details. An administrator who needs to know how things are likely to route would use "traceroute".

There are privacy issues here: the classic case is that a shelter for battered women must be able to block callers from knowing where the call really goes. (that was high politics during the introduction of caller ID) A proxy should be able to hide details of how it is handling a call at its end, and then it's up to the other participants in the call to decide whether to continue. Hiding switching inside a PC is easy anyway, so we can't stop it.

A grouping mechanism is probably more flexible than a strict hierarchy, and perhaps a better match to the logical structure. Figure 3 describes end-to-end encryption and hides the irrelevant details of transport over the rest of the network and of the processing required to get low-rate data suitable for encryption. Hierarchy (two levels, anyway) naturally describes the mapping from physical (less detail) to logical (more detail, because there are several computing task per computer and several logical links per T1) networks.

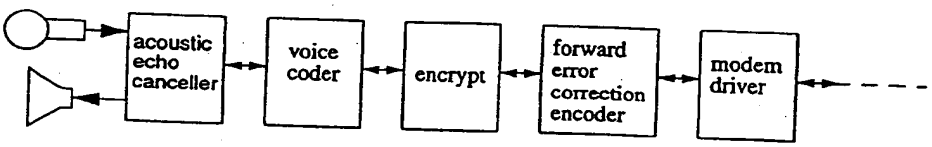


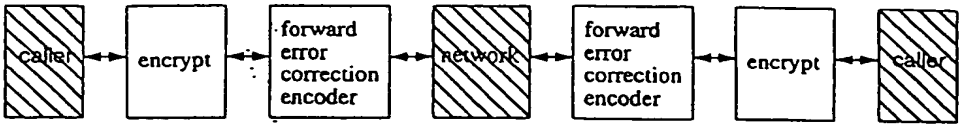
FIGURE 2. Part of the graph representing a simple call. The blocks are described in general terms in Section 5.2 below.

rules of composition would specify what kinds of networks make sense, and could be

1. left for the Java level to figure out
2. implied by a strong typing mechanism for ports
3. enforced by the objects themselves

Point 2 seems like a good option, but typing is rather dynamic: forward error correction can be applied to any type of data, and when inverted gives back the original type.

FIGURE 3. Representing end-to-end encryption; blocks marked "caller" and "network" are groups.



Not all the information flows in the signal path from one subscriber to another: there are also flows to and from the service provider's billing and management software and to the subscribers' call processing software. For example, if the number of uncorrected errors becomes excessive, it may be appropriate for the encoder to raise an exception in the call processing code so that a more robust one can be chosen. This same example also shows that it can be necessary to modify the graph while it is running.

Hiding the internals of the network as in Figure 3 is probably right most of the time, but not always. For example, a user may want to be sure that his data is never carried on a certain type of link (one belonging to a competitor, for example, or one for which availability is only statistically estimated) or that it travels by two totally independent paths through the network (for reliability). It is possible (likely?) that some detail will be hidden from the end user ("client") but not from the server.

5.1 Prior Art

Katosizer; Mauricio thesis?, Tornado...

5.2 Examples of signal processing objects

First let's look at some examples of signal processing objects, with an eye to understanding what requirements they will place on our system.

The signal path

linear and adaptive filters Classic linear filtering is used to remove DC and 60/120/180Hz tones from power-line interference and to smooth signals for downsampling; in a digital system the downsampling and filter are usually combined in a more efficient decimation or rate-conversion block. Other applications, standard in audio but rarer in telephony, include tone controls and generation of reverberation. Computation loads for simple filters are very small, on the order of 1-10 multiply-adds per sample (80kIPS), and completely predictable. If the processor on which a filter is running crashes and a new filter is restarted, there will be an audible "click" unless state is preserved, and the internal state may vary quite quickly.

The main requirement that filtering places on the system are:

- that multiply-adds at the 16-24 bit level be fast.
- that overheads for simple algorithms be small.

Adaptive filters tune their coefficients to the particular call in progress: the best-known case in telephony is the echo canceller, of which variants are designed to cancel acoustic echoes (resulting from an acoustic path from the handset's loudspeaker to its microphone) and electrical echoes resulting from system components (specifically the "hybrid" that converts from 2-wire to 4-wire). An echo canceller is typically a transversal filter with a few hundred

taps (multiply-adds), supervised by code that tries to determine the appropriate order and to turn it off when it appears to be unwanted or diverging. Echo cancellers that attempt to deal with acoustic echo from a speakerphone in an office environment need thousands of taps and sophisticated update algorithms: this area is in flux. There are two types of state in an adaptive filter: current coefficient values and signals. Coefficients could be checkpointed from time to time, but it's more expensive with signals because they vary more rapidly.

companding techniques use "compression" algorithms that try to adjust gains (smoothly) so as to keep a signal's level more constant and "expansion" algorithms that adjust gains to exaggerate signal-level variations. Some techniques used in audio are frequency-dependent, such as Dolby companding which adjusts filter cutoffs to suppress background hiss when signal levels are low. An extreme example of expansion is "squelch" in which signals with power level below a certain threshold are turned off completely to minimize idling noise. In telephony the most common variant is "echo suppression" (as opposed to :"cancellation") in which the signal path from the quieter user has its gain reduced — which reduces the loop gain for echoing and feedback oscillation. Companders use around 5-50 operations per sample.

Instantaneous companders work on a sample-by-sample basis, and the common A-law case is covered under "coders" below.

Echo suppressors cause trouble for modems, so there is a convention of watching for a 2100Hz tone at the beginning of a call as an instruction from a modem to disable echo suppression.

A 3-way combiner takes three input signals and produces three outputs: in principle "C" gets voice "A+B", "B" gets "A+C", etc. The idea extends naturally to N-way combining. Companding (above) is used to improve subjective quality by suppressing noise from inactive channels.

One vertical application that we've discussed would leave the source channels uncombined, so that 3-way calls can be taken in stereo and the various voices more easily distinguished. This is mostly an application at the end-user's PC, because that's where the stereo speakers are. Our contribution will be to provide the packets in a timely fashion.

voice coders are used to reduce the bandwidth requirements for voice signals. There are many types. but broadly they can act on the waveform, minimizing some mathematical measure like error power; they can model the source; or they can model what the ear will notice. Coding for compression is an active research area, and we have to assume that a steady stream of new coders will appear.

"Telephony classic" uses waveform coding in the form of 8kHz A-law (or μ law). Sampling is done at 8kHz on a signal filtered to pass the range from 300Hz to 3300Hz. The passband was defined to get good subjective scores on speech quality and intelligibility, and the sample rate is designed with a 33% margin over the Nyquist minimum in a trade-off between network and prefilter costs. A-law and μ-law are specialized 8-bit floating-point representations, chosen as a way to get roughly constant signal-to-noise over a wide range of signal levels. By comparison, CD sound is stereo 16-bit fixed-point sampled at 44.1kHz — needing roughly 24 times the bandwidth, i.e. T1. Because speech varies slowly from sample to sample, the same quality can be had for roughly half the bandwidth with ADPCM which (roughly speaking) digitizes the derivative instead.

Most digital cell-phones use a variant of linear prediction coding, which tries to model the incoming sound in terms of a sound source that simulates the vocal cords or airflow and which in turn drives a filter that models the larynx. This requires less bandwidth than waveform coding because the larynx moves more slowly than the waveform, but works badly for anything other than speech or even for speech in a noisy environment. These "source coders" are an active topic of research and currently produce tolerable speech at output rates anywhere from 4kb/s up. A typical

modern coder uses about 50MIPs of DSP capacity. Coders typically operate on 20msec frames of data, and hence add at least that much delay to the signal path.

Source coders typically try to detect silence, and avoiding the transmission of silence typically saves about 50% of bandwidth on average. At the decoding side it is conventional to replace silence with "comfort noise" so that the listeners know the connection is still live.

Source coding is hard to use for music, because it would be necessary to model a large number of different instruments alone and in combination, so early digital audio (e.g. CD and DAT) just used waveform coding with enough bandwidth and dynamic range to satisfy (more or less) the ear. Minidisc and digital compact cassettes brought in coding that reduced CD bandwidth by a factor of about 10 by using psychoacoustics. (in particular masking effects, where loud tones mask nearby ones for normal ears, and bandwidth can be saved by not transmitting the inaudible components) This type of technique can also be rate-adapted (as in RealAudio) and is a good candidate for high-quality speech in our networks.

Conventional filters, companders, etc. won't work on a coded signal, so it's standard to decompress before filtering. In some cases we may be able to avoid this, though: for instance N-way combining can take advantage of silence to do companding "for free", and only needs to decode and recode during double-talk.

MPEG (Motion Picture Experts Group) coders do the same type of thing for video signals that perceptual coders do for music. Components of a video stream at high spatial frequencies are digitized at low resolution, (using 8*8 discrete cosine transforms to do the filtering) and "motion estimation" is used so that components of an image that can be derived from adjacent frames are not retransmitted. We should probably leave MPEG for the end-user's PC, because it is very demanding and because specialized hardware exists for it, so we are likely more interested in its traffic properties. Straight digitized TV costs roughly 30frames/sec*200kpixels/frame*3colours*8bits/colour, for 144Mb/s. That's beyond what 3G wireless is built to handle, but MPEG2 gives similar quality at 2Mb/s — hence the 3G requirement for that rate. MPEG2 is bursty, needing more capacity when the image changes suddenly.

At the low-quality end, videoconferencing is usually done at 128kb/s. At this rate the coding process adds hundreds of msec of delay and the picture is poor.

- if we get a lot of demand for full-motion video, then 5MHz slots won't cut it. 20MHz slots and generous use of antenna diversity could support 10-40 users at that rate.
- we may want our NOS to be able to initiate processes in (colonize?) the end-user PC so that video services can be set up easily.

voice-mail and its video- and text equivalents are usually thought of as pure data, but should be seen as objects with methods for reading and writing, or as filters that persist after calls complete. The reason to generalize is to allow different types of coders (and encryption, and fax data, etc.) to be used in a flexible manner with voice-mail.

Reading voice-mail can be thought of as accepting a call—albeit a time-shifted one. Voice-mails are all pending requests to the called party's proxy, and display the graph of the call that set them up (even though it has long since been torn down) so that the accepting party can see data type, coding and encryption needs, source of call, check who is paying, [1]etc.

1. Note the odd possibility of reversed charges in voice-mail. The calling party still has to pay a deposit for disk usage etc., in case the called party refuses to pay. This is a good tough use case for good billing software.

A single-use proxy stays alive and attached to the mail, into which you may also call to retrieve mail. This permits group messaging or retrieval by password, situations in which the voice-mail doesn't know who to contact. In this sense reading voice-mail can be thought of as originating a call.

channel coding

FEC (Forward Error Correction) uses mathematical algorithms such as XOR convolutions between the data and a given sequence to produce redundant bits that can be used to detect and correct errors in transmission. For our wireless channels this will be very important. ARQ (Automatic Repeat reQuest) schemes like TCP are simple and efficient and can be arbitrarily reliable, but add variable latency—making them impractical for voice. In telephony, frames which can't be corrected using FEC are discarded, and a rate of 10^{-3} is considered acceptable. There are trade-offs among redundancy, power and error rate.

The brute-force (and not used) example of FEC is triple redundancy, in which every bit is transmitted three times, and the Hamming codes used in memory systems are also fairly straightforward. The algorithms used in cellular telephony typically have rates of 1/2 (half the bits are data) and are subtle to derive but cheap to implement in hardware. The operations involved are generally at the bit level. It is tempting to use FPGA techniques for these algorithms, but we will have to consider the trade-off with software complexity: open FPGA assignment would suggest that third parties can write VHDL. We would probably want to be "pseudo-open" on this, applying just enough of the open philosophy to make design simple and systematic for our own purposes but not necessarily allowing the unwashed to program the FPGA.

the pure Internet "dumb network" philosophy would use end-to-end error correction, but that is inefficient when a particular link is known to be unreliable. If the inefficiency is just that one packet in a thousand is carried a few extra T1 hops, it doesn't matter much, but FEC also increases (doubles, typically) packet size.

Interleaving involves distributing the bits that are protected by one FEC word over several packets, so that even if one packet is badly corrupted (a "burst error") no single FEC word will suffer more errors than it can correct. Interleaving adds substantially to latency. The computations are cheap, but again so bit-oriented that a hardware solution is tempting.

Encryption is needed over wireless channels for security, and may also be desirable end-to-end. Digital cellular systems do a mediocre job of encryption over the air, and immediately convert to cleartext at the basestation on the assumption that customers trust the telco and so as to make the signals compatible with the rest of the 'phone system.

Encryption is the subject of a lot of research and growing commercial interest, so we have to expect a steady stream of new software. The computational loads of some of the more exotic encryption system are fairly heavy, though "triple DES" (standard in the banking industry) is not too bad: a collection of a couple of dozen bit-shuffles and O(100) 4-bit table lookups.

Internet traffic will best be encoded in the user's PC rather than at our patchpoints, to minimize the amount of cleartext running around. Still, if our system is being used to implement a virtual private network the users may be transmitting cleartext around their Ethernets and using our system to bridge remote Ethernets—in which case we should do the encryption so that they have a "no-fuss" secure method.

We will have to use good encryption, including signature techniques, on our wireless control links.

modems

A user may choose to plug a modem into one of our RJ-11 jacks. For Internet use that would be perverse, because plugging into the Ethernet jack would be faster, but it may make sense for a fax machine. We could simulate a land-line, perhaps with ADPCM or even straight PCM, but a voice coder would destroy the data.

The right solution is probably to detect that the source is a fax, and implement a fax modem in software at the patchpoint and at the gateway closest to the receiver so that our system just needs to transmit the raw fax data. This can be used to economize on latency as well as on bandwidth, as long as the fax at the far end can be spoofed by our modems.

IP fax is seen in the industry as an easy win, because fax traffic rivals voice traffic on volume, so we should do this fairly early. We may want a partner to contribute the fax data pump.

voice mail

Current telephony practise for voice-mail is to convert calls to high-rate PCM, ship them near to the intended recipient at high bandwidth and with the usual telephony low latency, then voice-code them (to save space) and stick them on disk.

We can save on network load by leaving calls coded as for the wireless link, and by accepting long latencies (best-effort service, say) for coded packets. We can leave the voice encrypted. We can use disk space wherever it is available, though it is best to move it in advance close to the most likely place from which it will be read — because latency is less tolerable when listening to voice-mail.

Integration of a user's various mailboxes (e-mail, voice-mail and fax) is a current industry trend and a nice vertical application for someone to develop. It is a big project to do right, since it may involve text-to-speech and vice versa.

links between the signal and control paths

DTMF signalling is the familiar "touch-tone" technique of using a pair of tones ("Dual Tone") each at one of four frequencies ("Multi Frequency") to signal switches for dialling and end-user equipment for voice-mail hell. The encoders can be implemented with a simple filter or a table look-up arrangement, and the receivers can be implemented with a group of filters and slicers at roughly 30-100 operations/sample.

pulse dialling detection involves counting strings of open-circuits on the 'phone line at about 10Hz. The "flash" or "link" buttons often used to signal the desire to set up a 3-way call basically dial "1".

Both of these filters provide inputs to call processing software from the signal path. That path doesn't have tight latency requirements, unless you want to suppress the DTMF in the path to a called party.

voice recognition can be used to replace dialling and to offer more sophisticated call control from a conventional telephone, or to do authentication. Computational loads can be quite large (larger than voice coding, for example, which is typically a component of voice recognition) and the area is still subject to active research. Computational loads can also vary widely as a function of the input data.

Ideally, voice-operated services can be speaker-independent, but systems that can handle large vocabularies generally have to be trained for the speaker. Voice authentication systems obviously need training. The need for speaker-dependent state suggests that these algorithms will need access to a library (which they may also want to update if they are capable of continual retraining).

The heavy loads, the use of disk resources, and the fact that voice coding is typically the first step combine to suggest that a typical voice recognition application would do voice coding on the patchpoint but might do the rest of the crunching on a compute server at the basestation side. The right voice coder for recognition isn't necessarily a standard one for wireless, so we might have the interesting situation that choosing to use voice recognition somewhere in a call graph constrains the voice coder elsewhere. Obviously the call setup agent could just ship both kinds of data, but that would be an expensive use of the wireless resource.

call progress tones like dial-tone and ring and busy signals give call-processing software a way to drive the signal path. Modern systems also allow the use of speech clips, although this brings up some interesting questions of how to handle multilingual environments. Presumably a language preference is part of a user's state.

Users who have a good display device available would rather use it than hear ringing tones. This is an interesting example of the need to be able to abstract part of call processing — we'd like to be able to "plug in" arbitrary ways of notifying the end user that a line is busy without changing the rest of a piece of call processing software.

IP packets to and from the Ethernet port

We want to be able to filter IP packets that come in from the Ethernet port, too. There's not necessarily any warning that IP packets are coming, since IP is connectionless, but that just means that IP filtering is set up by default.

IP classifiers assign different types of traffic different priorities. They take a single input (unclassified IP) and produce multiple output streams. We'd need to classify packets before they go over the expensive wireless link so as to implement the user's own policies on what to pay premium rates for. A default classifier might (for example) assign a lower priority to Web traffic than to IP telephony, or give one particular Ethernet source higher priority than others. Classifiers belong at the network input and perhaps also at both ends of the wireless link.

IP classifiers can also manage traffic in the other direction, regulating flows onto the Ethernet from the patchpoint.

traffic shaping, traffic policing and radio resource management go together for IP packets. Traffic shaping uses leaky buckets (etc.) to force traffic statistics to match the profile promised in a QoS negotiation; traffic policing is the same thing when done by trusted code at the input to a network. We'd presumably have a standard "traffic cop" filter installed as part of any IP path. This is an example of a parameterizable filter, with cell rates and bucket sizes as parameters.

Because IP flows can be very bursty, we may need to allocate new radio channels/slots when a queue starts to build up, then give them back when it empties. In the absence of traffic (when its queue has been empty for a length of time comparable to a channel setup delay, for example) we'd assign a stream to a collision-detect radio channel. The channels are radio resources, and management policies are needed to share them efficiently among data flows. (including flows among unrelated but nearby basestations and patchpoints)

header compression is used to avoid sending 40 bytes of IP headers over the radio channel. RTP is one standard, and circuit switching can be regarded as an extreme case of header compression—source and destination addresses are known at channel setup.

Compression techniques are particularly sensitive to state: if the decompression filter at the other end of the link crashes and restarts with old state, traffic may be permanently misdirected. There are three obvious philosophies for dealing with this: checkpointing critical state information; adding a global "reset" signal that is passed to all filters after a crash recovery and which could cause resynchronization; and adding a private "restart" signal between the compression and decompression filter.

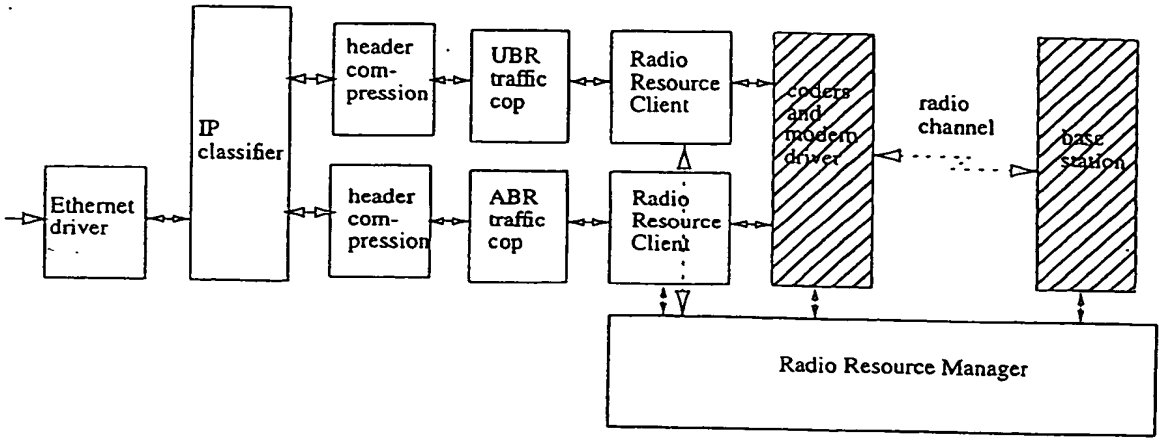


FIGURE 4. IP packets originating from the Ethernet port are classified, compressed, shaped/policed, and then handed to radio resource management software that runs across the radio link.

Passing state between compression/decompression pairs of filters is a problem in general, not just for crash recovery, because state information may have to be passed at a higher level of reliability than the rest of the data. This suggests that there should be a "side channel" set up between pairs, as shown in Figure 5.

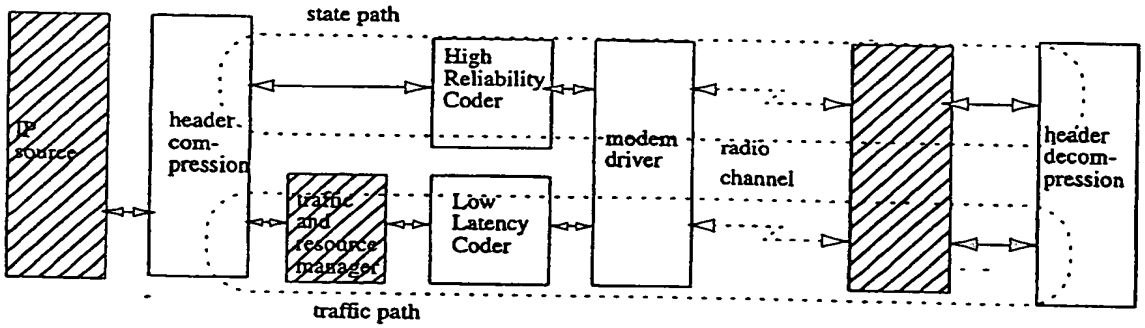


FIGURE 5. There is a need for reliable side-paths for state information.

firewalls implement security policies that control information flows between the inside and outside worlds, for example by forbidding telnet sessions. Firewall filters in our patchpoint would make for an interesting low-cost product, perhaps for small businesses.

gateways convert IP traffic to and from other forms, for example to PSTN voice traffic. Basestations with PSTN interface cards can serve as gateways, or we can use third-party gateways by adapting to their protocols.

caching of Web pages is a useful method of controlling bandwidth, but can be very tricky to implement because some Web pages are dynamic. In the case of the IBM/Kasparov match, for example, large volumes of traffic were generated for pages that changed on a timescale of minutes. An ISP could implement a cache to reduce the traffic onto the backbone while speeding up response for subscribers.

A cache shares information between users, so it doesn't necessarily show up directly in a user's call-graph. It logically fits as part of a gateway to the Internet backbone.

There are tricky legal issues, such as whether caching violates copyright; technical issues, such as whether caching makes cookie-based pages malfunction; and business issues such as whether it makes advertising banners and referrals miscount. Getting these things right makes caching difficult, but the performance win makes it worthwhile.

packet formatting e.g. for ATM and reassembly

figuring out tags; H.323

links between signal and OAM&P

Quality failures: queue overrun/packet drop; algorithm failure/residue; complaint

resource failures or shortages: mostly at the mapping stage but also: voice-mail overrun

links between signal and billing

resource use: PSTN, mailbox rentals, CPU,

900 numbers, e-payments

5.3 The signal-processing object

We will have a library of signal processing objects, instances of which can be wired together to set up communications graphs. These objects will have "ports" between which connections are made. What are the properties of the objects and their ports? Ports are part of a filter object, so we'll start there.

strongly typed ports

Ports should be strongly typed so that meaningless connections are difficult to set up: a voice coder that expects integer samples of a voice signal will do nothing useful if driven by the output of an FEC coder, for example. Presumably this also means that we have a library of "interfaces" (in the Java sense, i.e. types of ports) that new filters are expected to implement.

Ports may have a hierarchical structure; handshaking or backpressure signals, for example, may be associated with a data stream.

Ports usually have a direction to them (input or output), although they may have components that go in the opposite direction, as for example (again) when a handshake is involved.

Examples of port types can be found in the list of filters above, but let's look at them explicitly:

- sampled representations of audio signals: e.g. linear, A-law, ADPCM, samples of pre-emphasized signals. Ports of these types are also parametrized by sample rate, number of bits, and the characteristics of pre-emphasis filters.
- coded representations of signals, e.g. codebook-excited LPC. These can usually be parameterized (e.g. by filter length and frame and sample rate)
- alerts, which signal the occurrence of an event such as a hang-up or detection of DTMF, and reset ports
- billing ports, through which representations of money flow.
- parameter ports allow call-setup software to adjust such things as sample rates, or to read them.
- state input/output ports synchronize complementary pairs of coders and decoders.
- IP streams, and compressed versions (e.g. RTP streams)

ports and their types; e.g. A-law; alert; $; OAM&P; control

managing ports on queues

default port connections

resource requirements

resource requirements e.g. CPU average; CPU max?; disk; hardware (e.g. for drivers);

setting resource requirements

latencies

processing latencies

libraries

certification and privileges

certification and measurement philosophies

privileged/trusted filters needed for billing, OAM&P, NOS exec and scheduling. Must run on trusted hardware.

5.4 The link object

physical or virtual? It'll change.

characterized by BW & QoS

setting BW

latency, including queues

guaranteeing vs. measuring BW and latency

reliability

setting reliability; recommended coders

costing API (for service provider's pricing software)

billing and OAM&P ports

5.5 Call processing

5.6 Features to demonstrate

- role of demo vertical apps
- stereo 3-way
- "here I am" call forwarding; uses caller ID if possible to remotely reroute forwarding — needs authentication
- call forward that works through a switchboard by advising the caller of what's happening — or dials the extension
- e-cash calls

5.7 Project management

Bulletproof Java is a clearly definable separate project and a potential stand-alone product. If we can control it through licensing (cheap to everyone except competitors?), then we get a little bit of revenue at the same time that we create a community of software developers. We can hive the development and even some of the design off to TCS. Once it's there, we can move call processing that we have done in ordinary Java over to it and suddenly start to demonstrate survivability.

There are some high-level design trade-offs to think about. Ideally, a complete copy of the program state would be maintained on a backup machine at all times, but this would be extremely slow and synchronization with calls that change state of the network would be very tricky. Is it good enough to have the state of particular (programmer-define) objects protected?

6.0 Call Processing: Servers, Agents and Managers

intro: what is call processing? Setup, monitoring, teardown

clarity for individual

performance for system

6.1 Proxy

The key feature we need for call processing is that each end user is represented by a "proxy" in our system, which deals with

- incoming calls, deciding which ones get to ring a 'phone, which get busy signals, which go to voice-mail, etc.
- billing, representing the end user's policies on what to pay (900 calls? disk space for voice-mail?) and how. (bill me or Visa?)
- outgoing calls, using special "speed-call" directories or voice-dialling software, etc.

Proxies can also represent groups of users ("the operator", "911 dispatch", etc.). Generally, anything that can pay bills can have a proxy. It is the responsibility of the proxy to decide which telephone to ring.

Users outside our system will need "generic proxies" representing legitimate uses of the PSTN, for example, or to represent a PSTN calling party's intentions (caller ID, for example) to a WST proxy. A WST user who "roams" into the PSTN will need to be able to inform his/her proxy of how he/she can be found.

Well-known telecom services serve as "use cases" for a call processing architecture. A telephone call may be redirected by call forwarding or (on busy or no answer) to voice mail, for example; its billing may be redirected by use of 800 or 900 numbers; and during a call a third party can be added to set up a conference call. In a cellular system, users must be paged over a large area to find them for a call, and incoming calls may be redirected (by means of the "home location register" and "visitor location register" databases) to remote locations. The user's routing model for simple Internet packets is as simple as that for a 'phone call, but advanced IP features such as multicast, QoS control and mobility again bring in call setup and billing concerns.

In a telephone system, call features may conflict (busy wait and voice-mail features have different responses to a busy signal, for example). A proxy architecture takes care of this in the sense that a program has to be written to do one thing or another when sensing "busy". The difficulty will return if we try to implement a feature-level description language for call processing.

We're assuming that call processing software will be written in Java; key requirements for the language are:

- excellent security
- a large community of experienced developers
- object orientation
- simple net-based distribution mechanism

Proxies must persist in the presence of patchpoint and basestation failures, so that (for example) a user's forwarding instructions don't get lost during a crash.

Directories can be implemented as distributed databases, for scalability, and custom versions can exist. A directory really belongs to a user or group of users, rather than to the network provider; in an open system, we want a private "yellow pages" to prosper, for example, and anyway we can't forbid it because it could be provided (less efficiently) from any IP address.

A "call manager" will be needed for setting up complicated types of calls. In setting up a three-way call, for example, the question arises of what to do when the originating party hangs up: does the call drop? If not, either the calling party may find itself handling a number of calls of which it isn't part or it will delegate the task to a call manager. The call manager mechanism can also be used to simplify the task of call setup for simple calls.

Proxy as state machine

Proxies define a user's policies for making and taking calls, and complex policies will entail complex software; but we want policies to be easily defined. The combination of complexity and ease of use implies that we will want to impose a structure on the coding of proxies that makes for clarity and allows some sort of intuitive interface. The sequence of actions involved in taking a call can be described by a state machine, and the state machine can be edited as a graph—one which is in turn responsible for setting up switching graphs.

Figure 6 shows the piece of a proxy that implements the standard dialling model, for example. This representation can be edited graphically, and different blocks substituted to allow voice dialling or different interpretations of numbers. (e.g. the PBX convention "dial 9 to get out" or something where press-and-hold of a digit gives speed dialling)

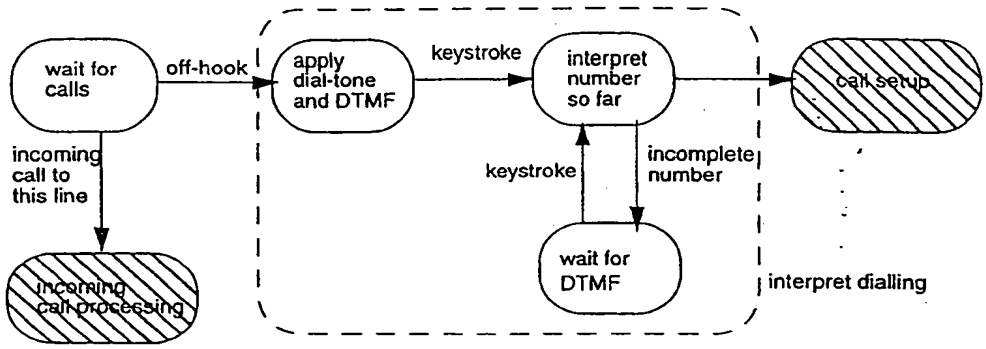


FIGURE 6. Section of a call-processing state-machine responsible for initiating calls according to a conventional dial-tone model

call setup is: set up your half the ideal way, ask a manager to get you a connection to someone else; they respond with counterproposals, the manager adjusts your graph to work. You can catch changes, or just let them go through.

you could have a state machine that goes through a sequence of offers

monitoring is: you can catch signals if you like.

Negotiating a connection

E911

6.2 Directory Server

distributed nature; like DNS, maybe LDAP

World Cup tickets

6.3 Mapping Server

6.4 RFQ Server

IPR?

For telephony, we need a mechanism in which:

1. the call manager asks for a quote on the cost of service for its traffic load at a certain QoS. The RFQ mechanism may be quite subtle, specifying a price/performance trade-off, but must degenerate easily to standard models.
2. the network responds with an offer of a certain capacity and QoS at a price and with a warranty. This response should be as informative as possible: for example, a response to an RFQ asking for an impossibly low latency should respond with an offer at the minimum practical latency.
3. the call manager accepts the quote, or perhaps tries again at a lower bandwidth or QoS.

4. the network sets up channels and reserves bandwidth.
5. the call manager alerts the various parties to start talking. They handle ringing/busy etc. themselves, but signal the call manager on hang-ups, DTMF tones or hookswitch flashes, etc. The call manager is also signalled by the network on failures to achieve contracted performance, and may renegotiate rates while a call is in progress.
6. the call manager alerts the network that the call is over
7. the network tears down the call

IP is a special case of this, in which a low-QoS ("best effort") low-bandwidth (compatible with Aloha channels, for most users) path to arbitrary IP addresses is negotiated at power-up. The call manager monitors queue lengths on either side of the radio link (being signalled when queues start to fill, for example) so as to adapt its QoS requests to load and bandwidth availability in a price-sensitive manner. The call manager is also signalled when packets are dropped.

A more advanced IP strategy typically involves different QoS as a function of parameters that can be deduced from the IP header. (source and destination addresses, ports, TOS bits, etc.) In our system this is enforced by filter processes in the patchpoint and/or network that segregate the user's IP traffic into a number of logical streams, each of which can have its own call manager. This allows third parties to offer "Turbo Telnet", for example, whereas in a typical modern network the priority of Telnet is fixed centrally (and low).

IPR?

Call Graph as API for negotiation

For new services, we cannot assume that there is a single pipe at a given bandwidth and QoS involved in a call; for example, a 3-way video-conference call might have one of the branches operating as voice-only at a much lower rate than the video branches. Our API for negotiation has to capture the whole structure of the call. For this reason, and to avoid adding new constructs, we will use the call-graph itself as the key specification both of desired service and billing method.

Essentially, the call-manager software hands a "schematic" of the desired call to the RFQ server, which looks at the "traffic cops" and similar blocks to determine a price, then sets parameters in "teller" blocks that feed money into key components. The modified graph is then returned to the call-manager for approval.

Figure 7 shows the graph that a call manager might pass for an RFQ on a simple call with a given bandwidth. The switching sub-structure is enclosed in a high-level block, and a particular payment method is attached to the entire block, but with the per-minute rate not yet filled in; that's the job of the RFQ server.

For use as part of the RFQ process, the calling graph needs to be able to express anything of interest to the user or the network, including latency, frame error rate, and nature of warranties. These things are expressed by including policing blocks in the call graph that enforce or test for compliance. The policing blocks have to trusted, which is an additional task for our certification group. We also have to be careful that a spoofing attack doesn't undermine the integrity of billing.

latency cops specify allowable latency. If we have accurate time references available, (e.g. by using GPS) then the structure of Figure 8 could be used. A timestamp is added to outgoing packets (perhaps all of them, in the case of IP, or perhaps just those starting bursts, as in telephony), and checked at the receiver.

The latency cop can also implement the buffering needed to eliminate the variability of incoming delay, which is important in voice applications. In this situation, the RFQ process estimates the worst-case delays through the channel, then sets a parameter in the latency cop that sets its maximum buffer size and how full it should initially be before passing data on.

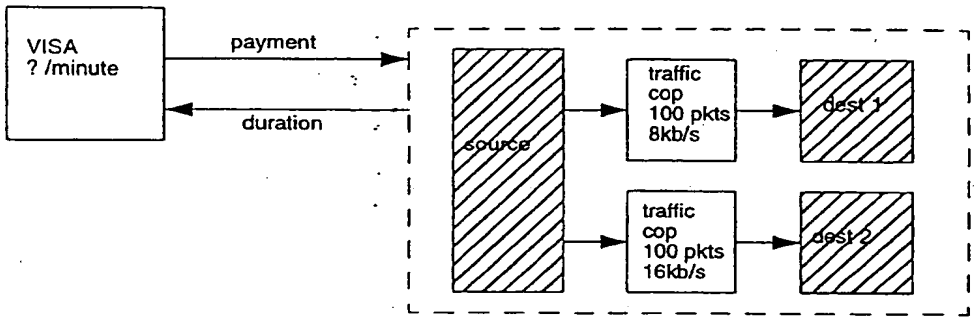


FIGURE 7. A simple call ready for a quote; this graph is passed to the RFQ server, which sets the rate (?/minute) and passes it back.

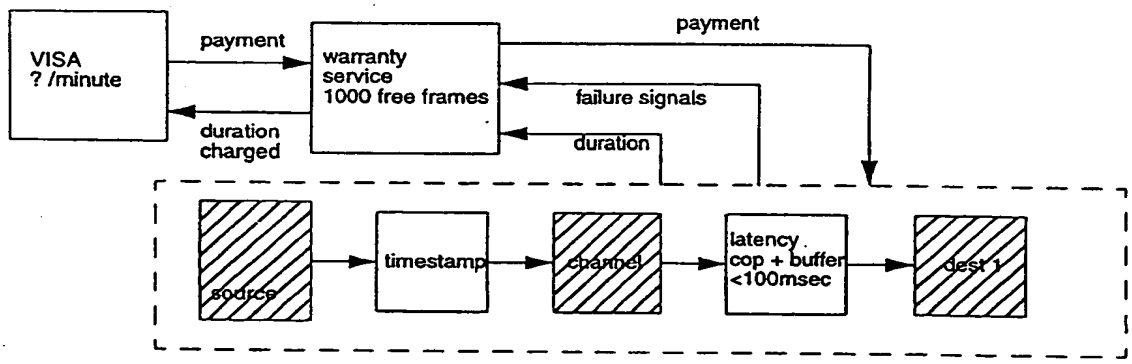


FIGURE 8. A call with a latency specification and a warranty.

When the latency cop detects a failure, it signals a warranty violation to the warranty service (Figure 8), which in turn reduces the time actually charged according to an agreed policy. The call manager sets up its desired policy, and the RFQ server may choose to charge a premium in order to assume that risk.

If absolute time references are not available, essentially the same structure can be used to measure and enforce round-trip delays. The policing function moves into the timestamp service, and an echo mechanism replaces the original cop. The echo mechanism may be rolled in with ARQ (e.g. for TCP) to save traffic.

Buffering to eliminate delay variability in each one-way path is more subtle in the case that absolute time references are not available, because the average input and output rates might not match. A "frequency-locked-loop" type of approach in which output sample rates are adjusted so as to keep a buffer size stable is used to solve the problem.

FER cops can be rolled in with error-checking at the destination, and work in much the same way as latency cops: when a frame error is detected, they signal a warranty failure.

detailed billing services are just interposed between the body of the call and the ultimate source of money (e.g. VISA), as shown in Figure 9. The call manager passes the billing service a copy of the entire call graph on call setup (so that it has all the necessary information to describe the call), and it also receives copies of warranty-failure alerts. It stores its results on rented disk, which can then be read through a Web interface or printed and mailed.

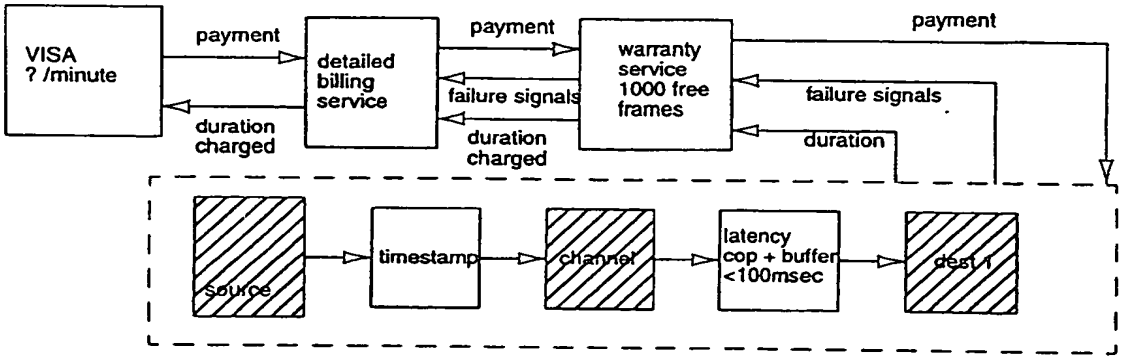


FIGURE 9. A call with detailed billing

multiple billing sources. are handled by attaching different money sources to different subgraphs, as in Figure 10. This will be useful in the case of point-to-multipoint services, like MBone or other pseudo-broadcasts.

976 calls or other calls that involve one user paying another are set up similarly: the payee includes a payment cop in a subgraph that will insist on money flowing in at a certain rate or on certain signals, and offers that subgraph to potential clients—who pay service charges both to the system and the payee, as in Figure 11. The RFQ process sets up these third-party flows in the same way as ordinary system charges. Because the 976 operator does not include a money source in the service subgraph, he can't be charged. This is also an example of a case in which the 976 service operator would not allow clients to descend the hierarchy and see the internal structure of the service (such as home 'phone numbers of the employees).

Robbie Q2

Telephony classic; one QoS, one price take/leave.

broker mechanism; brokers taking a cut.

charging for RFQs

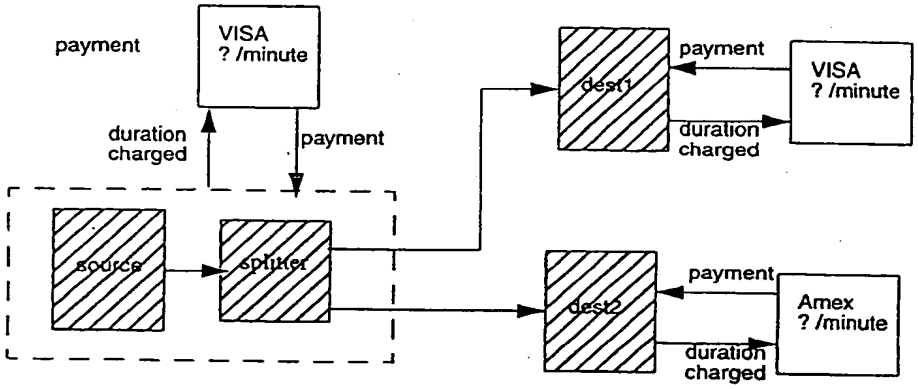


FIGURE 10. A call with multiple independent bills, e.g. for a broadcast-type service.

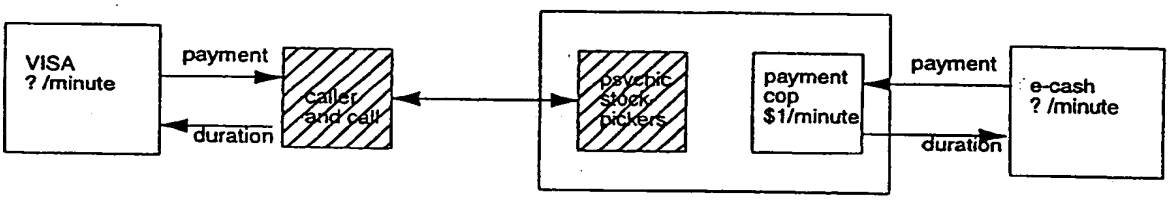


FIGURE 11. A 976 call. The 976 service provides a black box with voice and money ports, to which a caller attaches money and signals.

6.5 Link Manager

7.0 Implementation

Figure 12 describes a proposed structure for the software. A distributed communications substrate is interposed between user processes and the underlying machines, so that processes can generally be moved from one machine to another without being aware of it (either to distribute load or to recover from failures).

Processes running in the system come from different sources and accordingly get different treatment in terms of the trade-off between security and performance. Call-processing functions acting on behalf of the end users run in a protected "sandbox" environment on a virtual machine; those working on behalf of the network provider may run

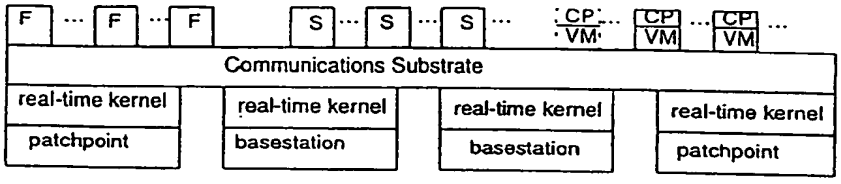


FIGURE 12. Structure of the proposed network operating system. Signals pass through filter processes "F", which also implement drivers and performance-sensitive functions on behalf of the network. Call processing on behalf of users is handled by "CP" processes running in a secure virtual machine ("VM") environment, which also includes checkpointing functions that can transfer control on failure to a "ghost machine" (shown as a dotted box for one of the call processing blocks). All these processes run on a common software communications layer, which places them on appropriate physical systems and arranges for their connections. Server processes ("S") also run on the communications substrate, but do not have the hard real-time constraints of the filter processes; secure call-processing functions are one type of server process.

there, but may also be implemented directly as processes running on the network operating system. User processes running as "filters", i.e. with the hard real-time demands that come from being in the signal path, also run directly on the communications substrate. Processes belonging to different users are protected from each other by the usual OS mechanisms (memory mapping, file privileges, etc.) but the source is also reviewed by WST. Filter processes on the same machine and part of the same call may share an address space and a thread of control, with data being passed with a function call mechanism and with connections to other hardware being handled by a stub that adapts a function call to a socket-type mechanism. These filters would still be dynamically linked, even with the functional mechanism.

Question for Mike and Mauricio: can we still share code segments among many users (20 people all using the same vocoder on a particular machine, e.g.) with this function-call mechanism? I guess with enough pointers and with functions mapped into different pages it should be fine and might cut down on I-cache misses? Is this IPR?

We have to shop around for the right kernel, but QNX is a candidate. A mix of business and technical issues should be considered:

- how quick is task switching?
- can we get the source, both to check for security holes and to extend it (for example to allow the scheduler to enforce advanced CPU-sharing policies).
- what per-unit price is the license, and what special conditions are involved (e.g. the Linux copyleft)

Migration done at exec() level? Daemons watch for things that need to migrate, either because watchdogs fail to bark or because queues start to fill up? Queue size anomalies reported to OAM&P.

processing objects: C code, may assume MMX (good for DSP, likely). Certification involves our seeing source, which is also escrowed to us so that failing third parties are not a problem for our users and so that we can coordi-

nate a recompilation/rewrite binge to a new CPU if necessary. Ports implemented a bit like sockets, but the system can retarget them (to a port on another process, including as a special case a driver to a network device). Ports better be fast. Backup of data that must survive crashes handled (in and out?) through a standardized port?.

8.0 IP portfolio

this section should outline our plans for patents, trademarks and copyright.

8.1 System patent

See "Telecom Network"

This protects a network of patchpoints and basestations, all running simple real-time kernels, all coordinated by a middleware layer that gives communications applications a view of the overall system where details of connections may be abstracted out.

8.2 Graph patent

See "Method for Configuring Communications Systems"

This protects the use of a graph structure to describe desired connections, for use in the overall system of Section 8.1.

8.3 Billing/RFQ patent

Patent sketch not yet written.

This protects the RFQ process: it's a server patent for the system of Section 8.1. We also protect the use of the graphs of Section 8.2 as an API.

claims

1. use of RFQ mechanism to permit third-party call setup
2. use of calling graph with policing functions to specify RFQ mechanism
3. brokers to translate between system and user pricing models, accepting risk

8.4 Mapping Server patent

Patent sketch not yet written.

This protects the mapping server. A system as in Section 8.1 having a process or processes that add functions (preferred embodiment as per Section 8.2) to permit a simplified description of a telecom "call", possibly including blocks representing QoS policing functions, to be mapped onto the physical network.

8.5 Middleware for Connectivity and QoS

Patent sketch not yet written.

Details on functionality of middleware layer used in Section 8.1. Is this a continuation?

8.6 Logo trademark

Telecom Network

patent application draft, Stumm, Snelgrove, De Simone

Assignee: WST

Appl. No.

Filed:

Foreign Application Priority Data

References Cited

U.S. PATENT DOCUMENTS

FOREIGN PATENT DOCUMENTS

OTHER PUBLICATIONS

ABSTRACT

A flexible telecommunications system and a method for allowing flexible and reliable configuration of same.

BACKGROUND OF THE INVENTION

Overall Background

telephone systems with "dumb terminals" + large legacy + ...: function too fixed, advanced features hard to do

Internet with smart but untrustworty terminals

variety of access methods

variety of switching standards

Desired Invention

Therefore, an invention which... is desirable.

SUMMARY OF THE INVENTION

should be organized by claim

Our system comprises

1. A collection of "terminals" (note: we need a better name, since "terminal" means the 'phone or the X terminal, and has to be the end of the line: patchpoints? On-ramps? TelePorts?) to which the subscriber connects telephones and networked PCs, and which have wireless links to...

2. A collection of basestations, owned by a service provider, interconnected by ...

3. ... a backbone network, such as an IPor ATM network or a collection of T1 lines.

4. A distributed operating system on all of the patchpoints and basestations, which supports ...

5. ... "filter" processes that can be moved between computers based on load, and that...

6. ... can also be moved on failure of a basestation or patchpoint, perhaps with loss of some process state and I/O packets, but without corrupting the structure of interconnection of I/O "ports" of the processes, which represents the logical behaviour of the system, and which is controlled by ...

7. ... a highly fault-tolerant Java-like (note: explain relevant Java features — sandbox, objects, API, download?) software layer that is so rigorously secure that it can be safely opened to third-party software developers.

Application

This is the architecture of our data-centric telephony system. Subscribers plug telephones and computers into the patchpoints, and software running on the patchpoint relays coded voice or data through a wireless modem back to the basestation, and thence through a collection of pieces of filtering software (such as coding translators, three-way-calling junctions, DTMF detectors, etc.) to other subscribers (or their software proxies).

Reliability is very important, hence the features relating to redundancy. There is implicit link redundancy through the use of a wireless standard that supports handoff.

Relation to Known Systems

Elements 1-4 could describe a particular topology of a networked operating system, with the relatively minor restriction that 1 and 2 have a wireless interconnection, and with a particular ownership structure. A network of servers with X terminals would look a lot like what we describe to that point. We will in any case wish to extend our system to operate in a wired network and with different ownership structures, so these restrictions are not key.

Element 5 describes the addition that platform computing makes to SUN and NT operating systems. The filter & graph view of real-time computing (part of elements 5&6, but for a single-user environment with simplified resource management) was in the Katosizer.

Element 6 is unique. Redundant systems exist, but can only economically switch processes on failure between very tightly coupled machines. They are designed to operate with no loss of information, whereas we compromise on that to allow realistic performance with redundancy between loosely coupled machines.

The comment "perhaps with some loss ..." means

- that we can generally tolerate "clicks" etc. that result from losing packets or having filter states change discontinuously, but we cannot tolerate an error in state that would result in the remainder of the call being useless — for example loss of the codebook in an adaptive vector quantizer. Applications that cannot tolerate errors will have to use expensive backup mechanisms and multiple paths.

- that we therefore have a mechanism in the filter API that allows backup across the network of state information that must be preserved (keepSafe(Object o) ?), and a related mechanism that extends TCP so that we can be certain which packets have made it to the other end even when the originator fails during the wait. This will be IPR.

There will be a good deal of IPR in the definition of the filter objects and API, particularly in areas that allow us to make development of filter software as open as possible.

Element 7 is of central importance for us, since it is what makes it possible for a small company to compete with the majors. There will be lots of nuts-and-bolts IPR, but we need to do a very careful job of defining and protecting it.

BRIEF DESCRIPTION OF THE DRAWINGS

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS OF THE INVENTION

A collection of computers interconnected by a network, some of the computers being connected to telephony hardware (including anything with microphones, loudspeakers, possibly LCD or video displays and possible videocameras and ...), with a software system that enables software components that perform particular tasks (such as recoding, conferencing, etc.) to be interconnected and for the connections to be dynamically changed. Use of sockets or function calls or threads or ....

WE CLAIM

Broad

1. Elements 1-6 of Section, "Summary of the Invention," on page 2

2. Elements 1-7 of Section, "Summary of the Invention," on page 2

1. items 1-6 for telecom?

2. items 1-7 for telecom

3. 1-6 for telephony in particular

4. 1-7 for telephony in particular

Fair and Square

5.

Nuts and Bolts

6. f

DRAWINGS

Drawings describing the problem

Drawings describing the invention in general terms

Drawings for preferred embodiments

Method for Configuring Communications Systems

patent application draft, Stumm, Snelgrove, De Simone

Assignee: WST

Appl. No.

Filed:

Foreign Application Priority Data

References Cited

U.S. PATENT DOCUMENTS

FOREIGN PATENT DOCUMENTS

OTHER PUBLICATIONS

ABSTRACT

A structure for configurable software systems applied to telecommunications that allows a library of software components to be interconnected in order to obtain desired behaviours.

BACKGROUND OF THE INVENTION

Overall Background

Telecommunications systems need to process the data flowing through them in complex ways, often with processing occurring on computer systems separated both geographically and administratively. Many communications paths are simultaneously active, and the processing applied to the various flows of data changes frequently and in a wide variety of ways. The software needed to control these computer systems is generally large, complex and difficult to change.

When the data flowing through the system represents voice, such as in a modern digital telephone network, special processing must be applied to implement such features as three-way or multi-way calling, voice-mail, voice recognition and authentication, call waiting, encryption, voice coding and DTMF (dual-tone multi-frequency, tradename TouchTone) detection. For data applications in general, such as electronic mail, remote computing, file transfer between computers or Web browsing, there are needs for security functions such as firewalls and encryption as well as datastream functions such as traffic shaping, error handling, prioritization, caching, format translation and multicast.

While telecom systems are already complex, new services such as video telephony, Int rnet games, video on demand, Internet audio, remote collaborative work and telemedicine are appearing. These services will need new families of features to be overlaid on the existing network, rendering the software development task yet more complex.

The complexity of present telecommunications systems software, and the extensive interactions between its software components, makes the development of new features very difficult. Software development is therefore limited to a "closed" group of trusted developers, which reduces the talent pool available and shuts out developers with new ideas for niche markets.

application to the modeling of communications systems

application to the modeling of physical systems

applications to transaction processing and to computation in general (Mike: I think it's easier to make one patent and to arrange ownership of various markets through contract than to try to subdivide a patent, because the umbrella claims have to be smaller in the two-patent case: it will be easier to squeeze between the gaps between two umbrellas than to get through one big one.)

Desired Invention

Therefore, an invention which allows the flexible definition of the processing of signals or data streams in communications systems, computer systems and computer networks is desirable.

SUMMARY OF THE INVENTION

organized by claim

BRIEF DESCRIPTION OF THE DRAWING

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS OF THE INVENTION

WE CLAIM

Broad

1. Anything using graphs to connect pieces of software (need to cut that down a bit...)

2. A configurable software system comprising
a library of filter processes;
a software process capable of connecting them in a prescribed fashion; and
an operating-system kernel capable of executing the connected processes

3. the software system according to claim 2 implemented on a plurality of computers interconnected by a communications network, in which connected filter processes may either run on the same or on different computers.

Fair and Square

4. the software system according to 3 applied to the processing of data streams in a telecommunications network

5. connected processes able to run in the same address space

Nuts and Bolts

6. function-call model

7. insertion of drivers for remote connection

8. strongly typed ports

DRAWINGS

Drawings describing problem

Drawings describing the invention in general terms

Drawings for preferred embodiments

What is claimed is:

1. In a telecommunication network including a local client, a local server, a remote server and a remote client, said local server and said remote server being operable to communicate via a plurality of communication media, a method for selecting one of said plurality of communication media to carry a communication service between said local client and said remote client, comprising the steps executed at said local server of:
monitoring performance parameters of said plurality of communication media;
calculating expected performance of said plurality of communication media for handling of each of said various modes of communication; and
determining a usage rate and a warranty to offer to said local client for said communication service between said local client and said remote client.

2. A telecommunications network as claimed in claim 1, further comprising a third party server electrically connected to said local server, wherein said steps of monitoring, calculating and determining are executed by said third party server.

3. A telecommunications network as claimed in claim 1, further comprising the steps of:
monitoring operation of a communication service; and
responding to said warranty being violated by providing compensation to said local client.

4. A telecommunications network as claimed in claim 1, wherein said step of monitoring operation of a communication service comprises recording packet losses.

5. A telecommunications network as claimed in claim 1, wherein said step of monitoring performance parameters of said plurality of communication media comprises:
transmitting sample packets to predetermined destinations within the network; and
calculating the available quality of service.

6. A telecommunications system comprising:

at least one basestation;

at least one client access point, each of said client access points being electrically connected to a single one of said at least one basestation;

a backbone network comprising a plurality of communication lines, each of said at least one basestations being electrically connected to said backbone network; and

a distributed operating system on said plurality of basestations and said plurality of client access points, being operable to provide switching, billing, voice and data services over said telecommunications system.

7. A telecommunications system as claimed in claim 6, wherein at least one of said client access points is electrically connected to at least one of said basestations via a wireless local loop.

8. A telecommunications system as claimed in claim 6, wherein said distributed operating system is a graph design.

9. A telecommunications system as claimed in claim 6, wherein said distributed operating system is implemented as a middleware layer, allowing freedom for third parties to add operationallity to said telecommunications system.

10. In a telecommunication network including a local client, a local server, a remote server and a remote client, said local server and said remote server being operable to communicate via a plurality of communication media, a method for selecting one of said plurality of communication media to carry a communication service between said local client and said remote client, comprising the steps executed at said local client of:

selecting an available communication service on the basis of a usage rate and a warranty.

11. A telecommunications system comprising:

at least one basestation;

at least one client access point, each of said client access points being electrically connected to a single one of said at least one basestations;

a backbone network comprising a plurality of communication lines, each of said at least one basestations being electrically connected to said backbone network; and

agent software operating on said at least one basestation and said at least one client access point, for negotiating communication services over said backbone network.

12. A method of communicating voice over data network calls between three or more participants comprising the steps of:

for each participant, mixing voice streams of all other participants together to create a composite incoming voice stream; and

transmitting each said composite incoming voice stream, to the corresponding said participant.

13. A method of CDMA wireless transmission in which the most distant receiver is identified, allowing a reduction in multipath interference and an associated reduction in transmitting power.

14. A method of CDMA wireless transmission as claimed in claim 13, wherein said transmission is quadrature amplitude modulation.

United States Patent & Trademark Office

Office of Initial Patent Examination – Scanning Division

Application deficiencies were found during scanning:

☐ Page(s)_____ of No Declaration were not present for scanning.
(Document title)

☐ Page(s)_____ of _____ were not present for scanning.
(Document title)

☐ Scanned copy is best available.